Maximum likelihood estimation for reversible mechanistic network models

Jonathan Larson and Jukka-Pekka Onnela
Department of Biostatistics, Harvard University, Boston, Massachusetts 02115, USA

(Received 7 September 2022; accepted 25 July 2023; published 21 August 2023)

Mechanistic network models specify the mechanisms by which networks grow and change, allowing researchers to investigate complex systems using both simulation and analytical techniques. Unfortunately, it is difficult to write likelihoods for instances of graphs generated with mechanistic models, and thus it is near impossible to estimate the parameters using maximum likelihood estimation. In this paper, we propose treating the node sequence in a growing network model as an additional parameter, or as a missing random variable, and maximizing over the resulting likelihood. We develop this framework in the context of a simple mechanistic network model, used to study gene duplication and divergence, and test a variety of algorithms for maximizing the likelihood in simulated graphs. We also run the best-performing algorithm on one human protein-protein interaction network and four nonhuman protein-protein interaction networks. Although we focus on a specific mechanistic network model, the proposed framework is more generally applicable to reversible models.

DOI: 10.1103/PhysRevE.108.024308

I. INTRODUCTION

The landscape of network models is dominated by two major types: Statistical and mechanistic. Statistical models specify a likelihood for each instance of a graph, usually in terms of some sufficient statistic like the number of edges. For example, consider the $G(n, p)$ graph [1]. It has n nodes, and each of the $\binom{n}{2}$ pairs of nodes (also called dyads) is connected by an edge independently and with probability p. If we consider n to be known, p to be unknown, and x to be the number of edges in the observed graph, then conditional on x, the "location" of the edges (i.e., which dyads are connected) does not depend on p. Thus, x is a sufficient statistic for p, and since x is binomially distributed with parameters $\binom{n}{2}$ and p, we can write the likelihood for x and carry out the maximum likelihood (ML) estimation of p. In this simple case, the ML estimate of p is given by $\hat{p} = x/\binom{n}{2}$.

Mechanistic network models are harder to characterize with a likelihood. Each such model specifies a set of mechanisms by which a network grows and changes over time. Although these mechanisms may consist of relatively simple steps, each step may erase the evidence of past steps. For example, consider the duplication-mutation-complementation (DMC) model [2,3]. It was intended to model how a protein-protein interaction network evolves. In a protein-protein interaction network, each node represents a protein in an organism and two proteins are connected if they interact. In the DMC model, the gene encoding a protein is erroneously duplicated with probability q_m, and the duplicated gene produces an identical protein. Over time, due to evolutionary pressure, the duplicate genes mutate separately, leading to two new proteins that may interact with different sets of proteins. The original and duplicated proteins interact with probability q_c.

The DMC graph is similar to the $G(n, p)$ graph in that it is generated through a series of Bernoulli random experiments. It differs because the outcomes of those experiments cannot be determined just by looking at the graph observed at the end. The addition of each new node can drastically change the relationships among previous nodes, erasing the evidence of previous duplication, modification, and complementation events. We can write down a likelihood for any given graph, but this becomes intractable when there are more than a handful of nodes. This is typical of mechanistic network models, which may be analyzed using likelihood-free inference [4,5].

There is a growing literature on statistical inference and model selection methods for mechanistic network models. Different papers use different approaches for addressing the intractability of the likelihood: They compare summary statistics by considering a trade-off between their informativeness and computational cost [6]; approximate the likelihood of a class of duplication-attachment models using an importance sampling scheme [7]; seek to identify a change point in network growth mechanisms using a likelihood-based framework that assumes the history of the network is known [8]; make the likelihood tractable by modeling network formation through conditional multinomial logit models from discrete choice theory assuming that edge formation events are observed [9]; and infer the node history for a random tree growth model [10].

In this paper, we propose a method for conducting the maximum likelihood estimation of the parameters of growing mechanistic network models. We use the DMC model as a test case, and thus our goal is to estimate q_m and q_c from a single observation of the network.

II. MODEL

A. DMC graph model

The algorithm for generating a DMC graph with n nodes is as follows. We first begin with a seed graph. Throughout this paper, we use a single node as the seed graph. Repeat the following three steps until the graph has n nodes. **Duplication:** (i) Select a node uniformly at random; this will be the

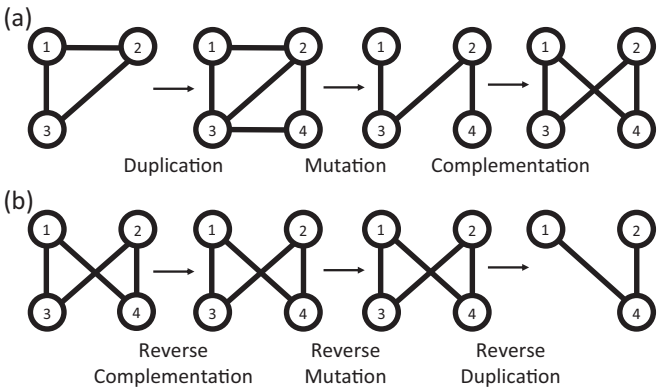

FIG. 1. (a) Schematic of the DMC model mechanisms, where node 1 is the anchor node and node 4 is the new node. (b) Schematic of the reverse DMC mechanisms, where node 3 is incorrectly assumed to be the new node and node 4 is incorrectly assumed to be the anchor node.

"anchor" node. (ii) Add a new node. (iii) Connect the new node to the anchor node's neighbors. **Mutation:** (i)"Modify" each of the anchor node's neighbors independently and with probability q_m. (ii) If a neighbor is modified, remove *either* the neighbor–anchor node edge *or* the neighbor–new node edge. Which edge is lost is determined by the flip of a fair coin. **Complementation:** (i) Connect the new node to the anchor node with probability q_c.

Figure 1(a) contains a schematic of the DMC model mechanisms. In the duplication step, node 4 (the new node) duplicates node 1 (the anchor node). In the mutation step, both of node 1's neighbors are modified, with node 2 losing its edge with node 1 and node 3 losing its edge with node 4. In the complementation step, node 4 connects to node 1.

Middendorf *et al.* [11] found the DMC model to explain the observed *Drosophila melanogaster* protein-protein interaction network better than six other candidate models. The authors argue that, in the modification step, it makes sense to remove either the neighbor–anchor node edge or the neighbor–new node edge, but not both, because each of these edges represents a function originally performed by the anchor node. If both disappeared, that would mean that both the gene coding for the anchor node protein and the gene coding for the new node protein mutated so much that neither could perform that function anymore. Since the function was presumably necessary, at least one of these edges must be maintained. The authors also note that the DMC model only allows for new edges in the context of duplication and complementation, meaning that mutations that result in brand new, advantageous functions performed by the protein are rare.

B. Graph deconstruction

Consider if we knew not just the (final) topology of the graph but also when each node entered the graph and which existing node served as its anchor node. Since each step of the DMC mechanism is reversible, we could run the evolution of the graph backward, determine the outcomes of the Bernoulli experiments, and estimate q_m and q_c.

Let G_n denote the observed graph of n nodes. Let $\theta_n = (\alpha_n, \beta_n)$, where α_n is an n vector containing the sequence in which the nodes entered the graph, and β_n is an n vector containing the sequence of anchor nodes. Thus the ith element of β_n was the anchor node for the ith element of α_n. (The first node, our seed graph, has no anchor node; if the nodes of the graph are labeled from 1 to n, we set the anchor node of the first node to be 0.)

Repeat the following three steps for $i = n, n-1, \ldots, 2$. **Reverse complementation:** (i) Set $W_i(G_n, \theta_n) = 1$ if the new node (the ith node in α_n) and its anchor node (the ith node in β_n) are connected, and $W_i(G_n, \theta_n) = 0$ otherwise. (ii) If the new node and the anchor node are connected, remove that edge. **Reverse mutation:** (i) Set $X_i(G_n, \theta_n)$ equal to the cardinality of the intersection of the neighbors of the anchor node and the new node. This is the number of unmodified neighbors of the anchor node. (ii) Set $Y_i(G_n, \theta_n)$ equal to the cardinality of the union of the neighbors of the anchor node and the new node. This is the original degree of the anchor node, before it was modified. (iii) Connect the anchor node to all the neighbors of the new node, and connect the new node to all the neighbors of the anchor node. **Reverse duplication:** (i) Remove the new node.

Here W_i, X_i, and Y_i are written as functions of G_n and θ_n because their values are completely determined by G_n and θ_n. Typically the true insertion order of nodes is, however, not known. Figure 1(b) contains a schematic of this process for θ_n', which is not equal to the true value θ_n. Here, node 3 is assumed to be the new node and node 4 is assumed to be the anchor node. In the reverse complementation step, nodes 3 and 4 are not connected by an edge, so $W_i(G_n, \theta_n') = 0$. In the reverse mutation step, nodes 3 and 4 have exactly the same neighbors, nodes 1 and 2, so $X_i(G_n, \theta_n') = Y_i(G_n, \theta_n') = 2$. Finally, in the reverse duplication step, node 3 is assumed to be the new node and is thus removed. Note that, for the true value of θ_n, $W_i(G_n, \theta_n) = 1$, $X_i(G_n, \theta_n) = 0$, and $Y_i(G_n, \theta_n) = 2$.

C. Inference

If we think of θ_n as a parameter like q_m and q_c, and we define

$$W(G_n, \theta_n) = \sum_{i=2}^{n} W_i(G_n, \theta_n), \tag{1}$$

$$X(G_n, \theta_n) = \sum_{i=2}^{n} X_i(G_n, \theta_n), \tag{2}$$

$$Y(G_n, \theta_n) = \sum_{i=2}^{n} Y_i(G_n, \theta_n), \tag{3}$$

then we can write the likelihood (likelihood function) as

$$\mathcal{L}(q_m, q_c, \theta_n) = q_m^{Y(G_n,\theta_n)-X(G_n,\theta_n)}(1-q_m)^{X(G_n,\theta_n)} \times q_c^{W(G_n,\theta_n)}(1-q_c)^{n-1-W(G_n,\theta_n)}. \tag{4}$$

The likelihood is defined as the joint probability of the observed data as a function of the model parameters. In practice, it is common to omit constants that do not depend on the model parameters, in which case the likelihood is only defined up to a multiplicative constant of proportionality. For example, the binomial distribution with parameters

n and p specifies the probability distribution of the number of successes in a sequence of n independent experiments; the binomial likelihood function $\mathcal{L}(p|n, y) = \binom{n}{y} p^y (1-p)^{n-y}$ without the multiplicative constant would be written as $\mathcal{L}(p) = p^y(1-p)^{n-y}$. In the DMC model, the likelihood is a product of two independent binomial likelihoods, one for the mutation process and one for the complementation process, which is how we arrive at the above expression.

The most common approach to parameter estimation is to select parameter values that assign the highest probability to the observed data. This is known as maximum likelihood estimation. In our case, we can maximize this likelihood with

$$\hat{q}_m = 1 - \frac{X(G_n, \theta_n)}{Y(G_n, \theta_n)}, \tag{5}$$

$$\hat{q}_c = \frac{W(G_n, \theta_n)}{n-1}, \tag{6}$$

$$\hat{\theta}_n = \arg\max_{\theta_n} \mathcal{L}(\hat{q}_m, \hat{q}_c, \theta_n). \tag{7}$$

Right away, there are some caveats worth mentioning. First, different values of θ_n may yield the same likelihood for a given G_n, so θ_n may not be identifiable. Also, θ_n resides in a space of dimension $n!(n-1)!$, which increases rapidly with n. However, since the duplication step renders the new node identical to the anchor node, it does not matter which is labeled new and which is labeled anchor. Thus, the parameter θ_n really only needs to encode the sequence of pairs used to deconstruct the graph, of which there are

$$\prod_{i=2}^{n} \binom{i}{2} = \frac{n!(n-1)!}{2^{n-1}} \tag{8}$$

possible values. Even though this is less than $n!(n-1)!$, it grows rapidly with n. For the rest of this paper, for clarity, we consider $\theta_n = (\alpha_n, \beta_n)$ as having $n!(n-1)!$ possible values.

The idea of estimating the age of each node (that is, when each node entered the graph) is not new. Navlakha and Kingsford [12] coined the term "network archaeology" for this endeavor and proposed the following algorithm for estimating α_n.

(i) Select initial values of q_m and q_c.

(ii) For each pair of nodes (u, v) in the graph, do the following.

(1) Let $W(u, v) = 1$ if u and v are connected by an edge and 0 if they are not.

(2) Let $X(u, v)$ equal the cardinality of the intersection of the neighbors of u and v.

(3) Let $Y(u, v)$ equal the cardinality of the union of the neighbors of u and v.

(4) Using the initial values of q_m and q_c, compute

$$\mathcal{L}(u, v) = q_c^{W(u,v)}(1-q_c)^{1-W(u,v)} \times q_m^{Y(u,v)-X(u,v)}(1-q_m)^{X(u,v)}. \tag{9}$$

(iii) Select the pair of nodes (u, v) that maximizes $\mathcal{L}(u, v)$ and reverse the DMC mechanism using those nodes.

(iv) Repeat steps (ii) and (iii) until the graph has one node.

Navlakha and Kingsford [12] sought to estimate α_n, not (q_m, q_c), although they did suggest using likelihoods to select optimal values of q_m and q_c.

Arguably, θ_n is not a parameter but a missing random variable that does not depend on q_m or q_c. This framework lends itself to expectation-maximization (EM) [13], which maximizes an intractable likelihood for incomplete data (in this case, the graph) by using a tractable likelihood for complete data (in this case, the graph and the sequence of new and anchor nodes). EM also avoids the problems outlined in Ref. [14]. Let $Z_n = (U_n, V_n)$, where U_n is an n vector containing the sequence in which the nodes entered the graph, and V_n is an n vector containing the sequence of anchor nodes. Thus the ith element of V_n was the anchor node for the ith element of U_n. (The first node, the seed graph, has no anchor node, and we simply set its anchor node to 0.) Then the probability of observing G_n and Z_n is

$$f(G_n, Z_n; q_m, q_c) = \frac{1}{n!(n-1)!} q_m^{Y(G_n,Z_n)-X(G_n,Z_n)}(1-q_m)^{X(G_n,Z_n)} q_c^{W(G_n,Z_n)}(1-q_c)^{n-1-W(G_n,Z_n)}. \tag{10}$$

Then the EM Q function, which is the expected value of the log-likelihood function with respect to the conditional distribution of the unobserved node sequence given the observed graph G_n and the current estimates of parameters q'_m and q'_c, is given by

$$Q(q''_m, q''_c|q'_m, q'_c) = \sum_z f(G_n, z; q'_m, q'_c) \log f(G_n, z; q''_m, q''_c), \tag{11}$$

where the sum is across all $n!(n-1)!$ possible sequences of new and anchor nodes. Given the initial values q'_m and q'_c, the E step consists of calculating $Q(q''_m, q''_c|q'_m, q'_c)$ and the M step consists of finding

$$\arg\max_{(q''_m, q''_c)} Q(q''_m, q''_c|q'_m, q'_c) = \left(1 - \frac{\sum_z f(G_n, z; q'_m, q'_c)X(G_n, z)}{\sum_z f(G_n, z; q'_m, q'_c)Y(G_n, z)}, \frac{\sum_z f(G_n, z; q'_m, q'_c)W(G_n, z)}{n-1}\right). \tag{12}$$

These maxima serve as the initial values q'_m and q'_c in the next round of EM, and the process repeats until the Q function converges.

Regardless of whether we think of the sequence of new and anchor nodes as a parameter or an unknown random variable, it is not what we want to know. We are interested in estimating q_m and q_c. Thus, it might behoove us to average over all possible

values of θ_n. Taking our cue from the EM algorithm, we could use the estimates

$$(\hat{q}_m, \hat{q}_c) = \left(1 - \frac{\sum_{\theta_n} \mathcal{L}(\hat{q}_m, \hat{q}_c, \theta_n) X(G_n, \theta_n)}{\sum_{\theta_n} \mathcal{L}(\hat{q}_m, \hat{q}_c, \theta_n) Y(G_n, \theta_n)}, \frac{\sum_{\theta_n} \mathcal{L}(\hat{q}_m, \hat{q}_c, \theta_n) W(G_n, \theta_n)}{n-1}\right). \tag{13}$$

Of course, this all supposes that we can exhaustively search all possible values of θ_n (or Z_n). But this is infeasible for graphs with more than a handful of nodes. Thus, we need to find algorithms that can find reasonably good values of θ_n in a reasonable amount of time. We tested a variety of algorithms on DMC graphs generated with a variety of values of q_m and q_c. We also tested the best-performing algorithm on an empirical protein-protein interaction network as described below.

III. METHODS

A. Model-generated graphs

We generated small DMC graphs with 7, 100, and 200 nodes. For each of these three graph sizes, we generated 100 graphs, using every possible pair of (q_m, q_c) where $q_m, q_c \in \{1/11, 2/11, \ldots, 10/11\}$. We intentionally avoided the boundaries $q_m, q_c \in \{0, 1\}$ in order for the graphs to be random. For example, since we used a single node as the seed graph, setting $q_c = 0$ would have generated a graph with zero edges, and setting $(q_m, q_c) = (0, 1)$ would have generated a complete graph. We then ran each of the following deconstruction algorithms on each graph.

(i) True θ_n: This algorithm uses the true value of θ_n.

(ii) True New, Random Anchor: This algorithm uses the true sequence of new nodes but selects anchor nodes uniformly at random.

(iii) NK, True Initial: This is the algorithm from Ref. [12], described in the Introduction, using the true values of q_m and q_c as initial values.

(iv) Exhaustive: This is an exhaustive search of all possible values of θ_n.

(v) NK: This is the algorithm from Ref. [12], described in the Introduction, using all possible pairs of (q_m, q_c) where $q_m, q_c \in \{1/5, 2/5, 3/5, 4/5\}$ as initial values. The estimate $(\hat{q}_m, \hat{q}_c)$ that yields the highest likelihood is used as the center of a new four-by-four grid of values, this time with gaps of $\frac{1}{5^2}$ instead of $\frac{1}{5}$. This process of using finer and finer grids of values is repeated until the likelihood stops increasing.

(vi) NK + 1: This algorithm is the same as NK, except it uses one additional grid of values after the likelihood stops increasing. Due to time constraints, we only ran this algorithm on graphs with 7 and 100 nodes.

(vii) Minimize $Y(u, v)$: Given the current state of the graph, this algorithm selects the pair of nodes (u, v) with the lowest value of $Y(u, v)$, the cardinality of the union of the neighbors of u and v. [If multiple pairs share the lowest value of $Y(u, v)$, one of these pairs is selected uniformly at random.] It then reverses the DMC mechanism using that pair of nodes and repeats. It is based on the fact that

$$\frac{\partial \log \mathcal{L}(u, v)}{\partial Y(u, v)} = \log q_m < 0 \tag{14}$$

and thus $\mathcal{L}(u, v)$ is maximized by minimizing $Y(u, v)$. [The sign of the partial derivative of $\log \mathcal{L}(u, v)$ taken with respect to $W(u, v)$ depends on the value of q_c; the sign of the partial derivative of $\log \mathcal{L}(u, v)$ taken with respect to $X(u, v)$ depends on the value of q_m. Thus, neither of these seemed to be a good criterion for selecting the next pair of nodes to deconstruct the graph.]

(viii) Minimize $Y(u, v)$, then NK: This algorithm runs the Minimize $Y(u, v)$ algorithm and then uses the resulting estimates $\hat{q}_m$ and $\hat{q}_c$ as initial values for the NK algorithm. The resulting estimates are fed back into NK as initial values, and this process is repeated until the likelihood stops increasing.

(ix) 1 Random: Given the current state of the graph, this algorithm selects a pair of nodes uniformly at random and runs the DMC mechanism backwards.

(x) 100 Random: This algorithm runs the 1 Random algorithm 100 times.

Some of the algorithms (Exhaustive; NK; NK + 1; Minimize $Y(u, v)$, then NK; and 100 Random) yield multiple values of θ_n. For each of these, we selected the value of θ_n that maximized $\mathcal{L}(\hat{q}_m, \hat{q}_c, \theta_n)$. In order to take advantage of the data on several values of θ_n, we also conducted EM as described above, except we summed over the investigated values of θ_n (or z) and not all possible values of θ_n (or z). We used the maximum-likelihood values of $\hat{q}_m$ and $\hat{q}_c$ as the initial values. Similarly, we averaged across the investigated values of θ_n as described in the Introduction.

We computed 95% confidence intervals for each algorithm's estimates of q_m and q_c as follows:

$$\hat{q}_m \pm 1.96\sqrt{\frac{\hat{q}_m(1-\hat{q}_m)}{Y(G_n, \theta_n)}},\ \hat{q}_c \pm 1.96\sqrt{\frac{\hat{q}_c(1-\hat{q}_c)}{n-1}}.$$

For the algorithms that yielded multiple values of θ_n, we selected the maximum-likelihood value when computing the confidence intervals.

In addition to obtaining estimates of q_m and q_c from each algorithm, we also calculated Kendall's τ [15] to compare the true sequence of new nodes to the estimated sequence of new nodes. Kendall's τ considers each of the $\binom{n}{2}$ pairs of nodes in the graph. If the relative ordering of the nodes in the pair is correct in the estimated sequence of new nodes, this pair is considered concordant. If the relative ordering of the nodes in the pair is incorrect in the estimated sequence of new nodes, this pair is considered discordant. Kendall's τ is equal to the number of concordant pairs minus the number of discordant pairs, divided by the total number of pairs. If the estimated sequence of new nodes is in the exact right order, $\tau = 1$. If the estimated sequence of new nodes is in the reverse order, $\tau = -1$.

Since the duplication step of the DMC model renders the new node and the anchor node indistinguishable, we computed two different values. The Strict Kendall's τ assumes that each algorithm cannot distinguish between the anchor node

and the new node when deconstructing the graph and thus must select one of them at random to remove. The Lenient Kendall's τ assumes that each algorithm can tell which node in a pair is the anchor node and which is the new node and thus removes the new node when deconstructing the graph.

We generated four additional 100-node DMC graphs with the parameters

$$(q_m, q_c) \in \left\{\left(\tfrac{1}{3}, \tfrac{1}{3}\right), \left(\tfrac{1}{3}, \tfrac{2}{3}\right), \left(\tfrac{2}{3}, \tfrac{1}{3}\right), \left(\tfrac{2}{3}, \tfrac{2}{3}\right)\right\}.$$

For each, we ran the following algorithms: True θ_n; True New, Random Anchor; NK, True Initial; NK; Minimize $Y(u, v)$; Minimize $Y(u, v)$, then NK; and 100 Random. For each graph, we plotted the log-likelihood as a function of the estimates of q_m and q_c.

B. Empirical graphs

To test our approach on an empirical network, we obtained the human protein-protein interaction network described in Ref. [16] from Ref. [17]. This is the largest and most recent protein-protein interactome obtained from humans. The obtained Human Reference Interactome (HuRI) graph contained 8272 nodes and 52 548 edges. We deleted 480 self-loops because the DMC model does not allow for them, yielding 52 068 edges. Then we sampled $p = 0.05, 0.10, \ldots, 0.90, 0.95$ of the nodes from HuRI uniformly at random and ran the Minimize $Y(u, v)$ algorithm on each induced subgraph. We also ran the Minimize $Y(u, v)$ algorithm on the complete HuRI graph.

We also obtained protein-protein interaction networks for the following organisms from the STRING database [18]: *Caenorhabditis elegans*, *Drosophila melanogaster*, *Escherichia coli*, and *Saccharomyces cerevisiae*. For each of these four organisms, we downloaded the dataset labeled "protein network data (scored links between proteins)," which contained only physical links. We removed self loops and links with combined confidence scores less than or equal to 700. Then we ran Minimize $Y(u, v)$ on the resulting graphs.

For the HuRI graph and the four STRING graphs, we sampled 10% of the nodes uniformly at random and ran Minimize $Y(u, v)$ on the induced subgraph. Using the estimates of q_m and q_c from each subgraph, we generated a DMC graph with the same number of nodes and ran Minimize $Y(u, v)$ on it as well.

We conducted all simulations on the O2 High Performance Compute Cluster, supported by the Research Computing Group, at Harvard Medical School. We conducted all analyses and simulations in R, Version 3.6.1 [19], except for the Minimize $Y(u, v)$ algorithm that we ran on the HuRI and STRING graphs. We conducted those analyses in PYTHON, Version 3.7.4.

IV. RESULTS

A. Model-generated graphs

The root mean squared errors (RMSEs) for q_m and q_c for model-generated graphs are in Table I. Minimize $Y(u, v)$ has the best performance (among algorithms that do not require prior knowledge of θ_n, q_m, or q_c) for both parameters on graphs of 200 nodes; and for q_c on graphs of 100 and 7 nodes.

TABLE I. Root mean squared error (RMSE) for q_m and q_c and for each graph size. For algorithms that yielded multiple values of θ_n (Exhaustive; NK; NK+1; Minimize $Y(u, v)$, then NK; and 100 Random), the results shown here are for the value of θ_n that yielded the highest likelihood. "N/A" indicates that this algorithm was not run on graphs of this size because of time constraints. The worst possible RMSE, achieved if $\hat{q}_m = 1$ whenever $q_m \leqslant 0.5$ and $\hat{q}_m = 0$ whenever $q_m > 0.5$, is 0.739.

	RMSE					
	7 Nodes		100 Nodes		200 Nodes	
Method	q_m	q_c	q_m	q_c	q_m	q_c
1	0.178	0.177	0.034	0.037	0.024	0.028
2	0.332	0.229	0.360	0.318	0.373	0.357
3	0.209	0.168	0.067	0.100	0.071	0.103
4	0.431	0.262	N/A	N/A	N/A	N/A
5	0.430	0.262	0.138	0.136	N/A	N/A
6	0.428	0.262	0.137	0.137	N/A	N/A
7	0.327	0.205	0.095	0.109	0.074	0.107
8	0.311	0.251	0.126	0.144	0.069	0.129
9	0.320	0.253	0.360	0.321	0.376	0.350
10	0.380	0.270	0.345	0.296	0.366	0.347

Its performance for q_m on graphs of 100 nodes is second only to Minimize $Y(u, v)$, then NK.

It is worth noting that the Exhaustive algorithm has the worst performance for q_m on graphs of 7 nodes. Table 1 in the Supplemental Material may explain why [20]. The likelihood is maximized when $\hat{q}_m$ and $\hat{q}_c$ are 0 or 1. If there exists a value of θ_n such that $\hat{q}_m$ and/or $\hat{q}_c$ is 0 or 1, the Exhaustive algorithm will find it. As Supplemental Table 1 shows, the Exhaustive algorithm estimates $\hat{q}_m$ to be 0 or 1 more than any other algorithm, and it estimates $\hat{q}_c$ to be 0 or 1 more than any other algorithm except 100 Random. The Exhaustive algorithm also has the greatest bias for the log-likelihood, meaning it is overestimating the likelihood more than any other algorithm. The table also contains the number of times each algorithm could not estimate q_m, which happens when $Y(G_n, \theta_n) = 0$. Since the Minimize $Y(u, v)$ algorithm minimizes $Y(u, v)$, it could not estimate q_m for the greatest number of graphs.

Table II contains the RMSEs for the EM and averaging results. For graphs of 100 and 200 nodes, performance does not improve, or it barely improves, when using EM or averaging instead of maximum likelihood. For graphs of 7 nodes, performance can be improved by using EM or averaging instead of maximum likelihood.

Observed coverage for the confidence intervals is in Table III. For every algorithm except True θ_n, coverage is low and gets lower as the number of nodes in the graph increases.

Average values of Kendall's τ are in Table IV. The Strict version is near zero for any algorithm that does not require prior knowledge of the true value of θ_n. When it comes to the Lenient version, 100 Random is the best-performing "naive" algorithm for graphs with 7 and 100 nodes; its performance is second only to 1 Random for graphs with 200 nodes. Running times for the algorithms are in Table 2 in the Supplemental Material [20].

TABLE II. Root mean squared error (RMSE) for q_m and q_c and for each graph size, only for algorithms that yielded multiple values of θ_n (Exhaustive; NK; NK+1; Minimize $Y(u, v)$, then NK; and 100 Random). "N/A" indicates that this algorithm was not run on graphs of this size because of time constraints. Abbreviations: Rnd, random; sqs, sequences.

	RMSE					
	7 Nodes		100 Nodes		200 Nodes	
Method	q_m	q_c	q_m	q_c	q_m	q_c
Exhaustive Max	0.431	0.262	N/A	N/A	N/A	N/A
Exhaustive EM	0.438	0.271	N/A	N/A	N/A	N/A
Exhaustive Ave	0.364	0.256	N/A	N/A	N/A	N/A
NK Max	0.430	0.262	0.138	0.136	N/A	N/A
NK EM	0.432	0.262	0.137	0.136	N/A	N/A
NK Ave	0.353	0.252	0.134	0.135	N/A	N/A
NK + 1 Max	0.428	0.262	0.137	0.137	N/A	N/A
NK + 1 EM	0.429	0.263	0.136	0.138	N/A	N/A
NK + 1 Ave	0.362	0.253	0.128	0.135	N/A	N/A
Min $Y(u, v)$, NK Max	0.311	0.251	0.126	0.144	0.069	0.129
Min $Y(u, v)$, NK EM	0.312	0.250	0.126	0.144	0.069	0.129
Min $Y(u, v)$, NK Ave	0.310	0.250	0.125	0.142	0.068	0.128
100 rnd sqs Max	0.380	0.270	0.345	0.296	0.366	0.347
100 rnd sqs EM	0.393	0.267	0.346	0.297	0.366	0.347
100 rnd sqs Ave	0.363	0.248	0.346	0.297	0.366	0.347

Figure 2 displays log-likelihood as a function of $\hat{q}_m$ and $\hat{q}_c$ for four additional DMC graphs. Each graph has 100 nodes and $q_m, q_c \in \{\frac{1}{3}, \frac{2}{3}\}$. The log-likelihoods attained for each graph vary, but the estimates appear in a characteristic oval pattern. The algorithms appear to perform the worst on the graph with true $(q_m, q_c) = (\frac{2}{3}, \frac{1}{3})$, even though it has the highest log-likelihoods. However, Figure 3 seems to indicate that, when the Minimize $Y(u, v)$ algorithm is used on 400-node graphs, the worst error for $\hat{q}_m$ should occur when q_m is high and the worst error for $\hat{q}_c$ should occur when q_m and q_c are both below 0.6.

TABLE III. Observed coverage for nominal 95% confidence intervals. "N/A" indicates that this algorithm was not run on graphs of this size because of time constraints.

	Coverage					
	7 Nodes		100 Nodes		200 Nodes	
Method	q_m	q_c	q_m	q_c	q_m	q_c
1	0.68	0.73	0.96	0.96	0.97	0.98
2	0.55	0.79	0.06	0.24	0.05	0.17
3	0.55	0.72	0.60	0.59	0.36	0.48
4	0.28	0.60	N/A	N/A	N/A	N/A
5	0.30	0.60	0.53	0.51	N/A	N/A
6	0.30	0.60	0.54	0.48	N/A	N/A
7	0.70	0.73	0.40	0.54	0.34	0.39
8	0.49	0.62	0.57	0.44	0.38	0.41
9	0.60	0.75	0.05	0.23	0.04	0.18
10	0.40	0.58	0.07	0.24	0.06	0.21

TABLE IV. Kendall's τ averaged across all graphs of a given size. The Lenient Kendall's τ is calculated as though each algorithm always removes the older node in a given pair when deconstructing a graph. The Strict Kendall's τ is calculated as though each algorithm must choose which node in a given pair to remove at random when deconstructing a graph. "N/A" indicates that this algorithm was not run on graphs of this size because of time constraints.

	$\bar{\tau}$					
	7 Nodes		100 Nodes		200 Nodes	
Method	Strict	Lenient	Strict	Lenient	Strict	Lenient
1	0.209	1.000	0.330	1.000	0.320	1.000
2	0.350	1.000	0.331	1.000	0.339	1.000
3	−0.025	0.533	0.039	0.240	0.056	0.248
4	0.006	0.556	N/A	N/A	N/A	N/A
5	−0.011	0.536	0.044	0.251	N/A	N/A
6	0.058	0.567	0.044	0.255	N/A	N/A
7	−0.025	0.530	0.029	0.183	0.049	0.187
8	−0.025	0.526	0.030	0.218	0.055	0.223
9	−0.034	0.518	0.014	0.346	0.010	0.345
10	0.040	0.564	0.006	0.364	0.000	0.349

B. Empirical graphs

Results for the HuRI graph and subgraphs are in Table V. Results for the STRING graphs are in Table VI. The subgraph induced by sampling 10% of the nodes from HuRI uniformly at random is in Fig. 4(a). There appears to be one large component and many isolated nodes. The color of each node corresponds to the estimated order in which that node entered the graph. The nodes at the center of the large component are estimated to be the oldest, and the isolated nodes are estimated to be the newest. The correlation between the estimated order in which a node entered the graph and its degree is -0.613 ($p < 10^{-15}$), meaning early nodes have a high degree and recent nodes have a low degree.

The DMC graph generated with q_m and q_c equal to the corresponding estimates from the graph in Fig. 4(a) is in

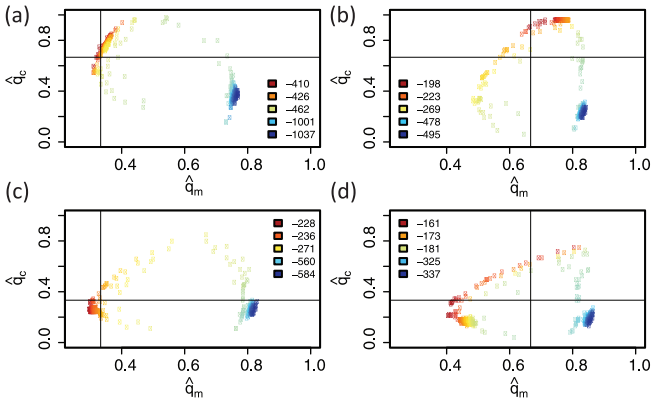

FIG. 2. Log-likelihood as a function of the estimates of q_m and q_c. The vertical and horizontal lines denote the true values of q_m and q_c, respectively, used to generate each graph. The legend for each plot displays the quartiles of the log-likelihoods and the corresponding colors.

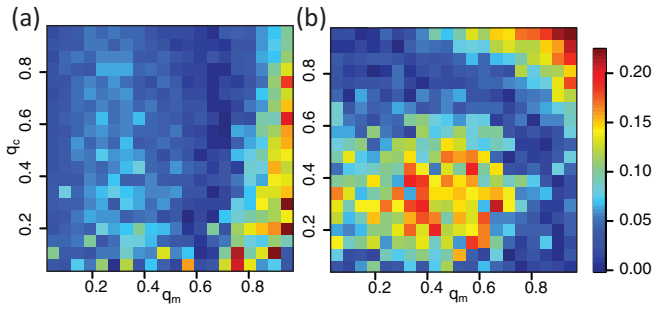

FIG. 3. Absolute error for the Minimize $Y(u, v)$ algorithm as a function of the true values of q_m and q_c for 400-node graphs for (a) $|\hat{q}_m - q_m|$ and (b) $|\hat{q}_c - q_c|$.

Fig. 4(b). There are many medium-sized components and many isolated nodes. The color of each node corresponds to the true order in which that node entered the graph. Old and recent nodes are evenly distributed among the medium-sized components and isolated nodes. The correlation between the true order in which a node entered the graph and its degree is -0.078 ($p = 0.03$), meaning early nodes have a high degree and recent nodes have a low degree, although the correlation is weak.

Figure 4(c) contains the degree distributions for the graphs in Figs. 4(a) and 4(b). The observed graph has more nodes of higher degree. Figure 4(d) is the same as Fig. 4(b), except with node color corresponding to the estimated order in which the node entered the graph. Here, we use the estimated order used to calculate the Strict Kendall's τ; i.e., we assume that the algorithm cannot tell which node in a pair is the anchor node and which is the new node and thus removes one at random in the deconstruction process. The Strict Kendall's τ for this estimated order is 0.016. The correlation between the estimated order in which a node entered the graph and its degree is -0.759 ($p < 10^{-15}$), meaning early nodes have a high degree and recent nodes have a low degree. Similar plots, but for the four STRING organisms, are in the Supplemental Material [20].

TABLE V. Results for the HuRI graph and subgraphs. The Minimize $Y(u, v)$ algorithm was run on only one core, in PYTHON. Here, p is the proportion of nodes sampled.

p	Nodes	Edges	$\hat{q}_m$	$\hat{q}_c$	Duration (h)
0.05	414	140	0.675	0.157	0.006
0.10	827	606	0.684	0.202	0.058
0.15	1241	1377	0.707	0.198	0.215
0.20	1654	2263	0.706	0.214	0.545
0.25	2068	3255	0.729	0.231	1.088
0.30	2482	4820	0.729	0.224	2.123
0.35	2895	5930	0.749	0.228	3.855
0.40	3309	8776	0.738	0.224	6.259
0.45	3722	10 679	0.743	0.224	9.238
0.50	4136	12 581	0.755	0.213	13.203
0.55	4550	15 987	0.749	0.218	18.639
0.60	4963	18 980	0.748	0.213	25.278
0.65	5377	21 340	0.751	0.224	32.951
0.70	5790	25 152	0.753	0.203	38.111
0.75	6204	30 631	0.755	0.209	58.366
0.85	7031	37 529	0.757	0.202	78.638
0.90	7445	41 985	0.758	0.210	94.396
0.95	7858	46 604	0.759	0.200	97.277
1.00	8272	52 068	0.760	0.204	112.806

TABLE VI. Results for STRING graphs.

Species	Nodes	Edges	$\hat{q}_m$	$\hat{q}_c$
C. elegans	5410	62 419	0.470	0.578
D. melanogaster	6439	90 385	0.430	0.621
E. coli	985	4223	0.257	0.646
S. cerevisiae	3557	50 198	0.341	0.708

V. DISCUSSION

Mechanistic models are useful representations of the processes by which networks grow and change over time. The DMC model, for example, has proved to be an accurate representation of how protein-protein interaction networks change over time [11,12]. However, the parameters of these models can be difficult to estimate using the maximum likelihood

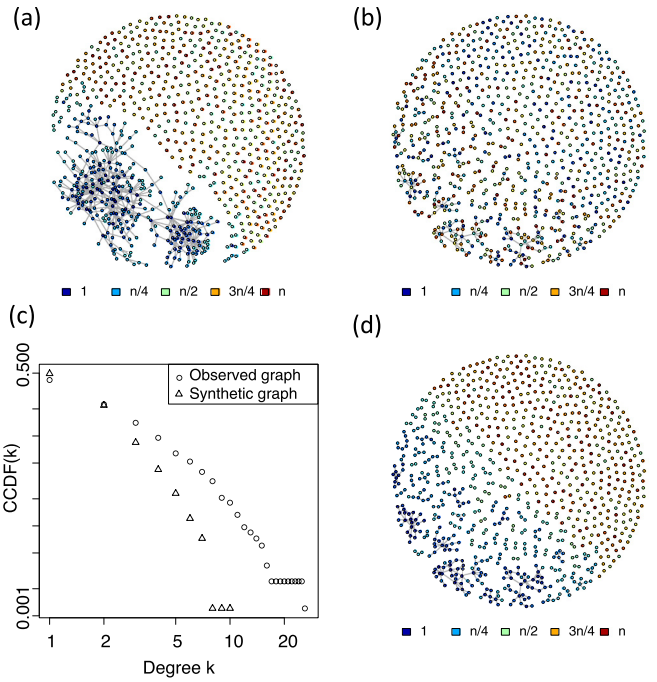

FIG. 4. (a) The subgraph induced by sampling 10% of the nodes from the HuRI graph uniformly at random. Node color corresponds to the estimated order in which the node entered the graph, with dark blue for the earliest nodes and bright red for the most recent nodes. (b) A DMC graph with the same number of nodes as the graph in panel (a), and with q_m and q_c equal to the estimated values of q_m and q_c for the graph in panel (a) ($\hat{q}_m = 0.709$ and $\hat{q}_c = 0.201$). Node color corresponds to the true order in which the node entered the graph. (c) Complementary cumulative distribution function (CCDF) of node degrees for the graphs in panels (a) and (b). (d) The same graph as in panel (b), but with node color corresponding to the estimated order in which the node entered the graph.

estimation. This paper proposed a framework for estimating these parameters and evaluated that framework with the DMC model.

When estimating q_m and q_c in a DMC graph, one can minimize the RMSE and time by using the Minimize $Y(u, v)$ algorithm. If the goal is to estimate θ_n, this is the worst algorithm. No matter which algorithm one chooses, a naive confidence interval will have low coverage. More research is needed to determine how to calculate better standard errors than those one would use for a sequence of independent Bernoulli experiments. Averaging across observed values of θ_n will probably not improve the estimates of q_m and q_c. The same can be said for using EM, although our results may arise from the fact that we used the maximum likelihood estimates as initial values for EM and did not perturb them. Using other initial values may yield a higher performance for EM.

For graphs with 7 nodes, the Exhaustive algorithm had the worst RMSE for q_m. This is probably an example of overfitting. Any time $\hat{q}_m \in \{0, 1\}$, the part of the likelihood containing $\hat{q}_m$ becomes 1, and any time $\hat{q}_c \in \{0, 1\}$, the part of the likelihood containing $\hat{q}_c$ becomes 1. It is more likely that one or both of $\hat{q}_m$ and $\hat{q}_c$ will be 0 or 1 for some θ_n in a small graph, and if such a θ_n exists, the Exhaustive algorithm will find it.

It is interesting to note that in Fig. 2 the estimates of q_m took the shape of an oval, with most estimates concentrated at the poles. This was the case even though many of those estimates arose from the NK algorithm, which uses initial values in a grid.

Further work can be done to make the algorithms proposed here run faster and to find other algorithms with better performance. But the main question that remains to be answered is whether the framework proposed here can be applied to other mechanistic network models. At first glance, it appears that this framework is restricted to reversible mechanistic network models. What happens if, for example, the mechanism involves deleting a node? Reversing it would require inserting a new node and connecting it to its former neighbors. But what if those neighbors cannot be determined from the current state of the graph? In this case, our framework can be extended by simulating multiple options, connecting this new node to neighbors selected through sampling, and then averaging across these simulations. Further research is needed to determine the performance of this extension.

ACKNOWLEDGMENTS

The authors thank Alessandro Vespignani, Edoardo Airoldi, and Rui Wang for their helpful suggestions. We also thank John Platig for suggesting the HuRI data. J.L. was supported by NIH Award No. T32AI007358. J.-P.O. was supported by NIH Awards No. R01AI138901 (Onnela) and No. R35CA220523 (Quackenbush).

[1] E. N. Gilbert, Random graphs, Ann. Math. Stat. **30**, 1141 (1959).

[2] A. Vázquez, A. Flammini, A. Maritan, and A. Vespignani, Modeling of protein interaction networks, ComPlexUs **1**, 38 (2003).

[3] A. Vázquez, Growing network with local rules: Preferential attachment, clustering hierarchy, and degree correlations, Phys. Rev. E **67**, 056104 (2003).

[4] S. Chen and J.-P. Onnela, A bootstrap method for goodness of fit and model selection with a single observed network, Sci. Rep. **9**, 16674 (2019).

[5] S. Chen, A. Mira, and J.-P. Onnela, Flexible model selection for mechanistic network models, J. Complex Networks **8**, cnz024 (2020).

[6] L. Raynal, T. Hoffmann, and J.-P. Onnela, Cost-based feature selection for network model choice, J. Comput. Graphical Stat. **0**, 1 (2023).

[7] C. Wiuf, M. Brameier, O. Hagberg, and M. P. H. Stumpf, A likelihood approach to analysis of network data, Proc. Natl. Acad. Sci. USA **103**, 7566 (2006).

[8] N. A. Arnold, R. J. Mondragón, and R. G. Clegg, Likelihood-based approach to discriminate mixtures of network models that vary in time, Sci. Rep. **11**, 1 (2021).

[9] J. Overgoor, A. Benson, and J. Ugander, Choosing to grow a graph: Modeling network formation as discrete choice, in *Proceedings of The World Wide Web Conference 2019*, WWW'19 (Association for Computing Machinery, New York, 2019), pp. 1409–1420.

[10] G. T. Cantwell, G. St-Onge, and J.-G. Young, Inference, Model Selection, and the Combinatorics of Growing Trees, Phys. Rev. Lett. **126**, 038301 (2021).

[11] M. Middendorf, E. Ziv, and C. H. Wiggins, Inferring network mechanisms: The *Drosophila melanogaster* protein interaction network, Proc. Natl. Acad. Sci. USA **102**, 3192 (2005).

[12] S. Navlakha and C. Kingsford, Network archaeology: Uncovering ancient networks from present-day interactions, PLoS Comput. Biol. **7**, e1001119 (2011).

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. R, Stat. Soc.: Ser. B (Methodological) **39**, 1 (1977).

[14] X.-L. Meng, Decoding the H-likelihood, Stat. Sci. **24**, 280 (2009).

[15] M. G. Kendall, A new measure of rank correlation, Biometrika **30**, 81 (1938).

[16] K. Luck, A reference map of the human binary protein interactome, Nature (London) **580**, 402 (2020).

[17] DFCI Center for Cancer Systems Biology, The human reference protein interactome mapping project (2021), http://www.interactome-atlas.org/.

[18] D. Szklarczyk, STRING v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets, Nucleic Acids Res. **47**, D607 (2019).

[19] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing (Vienna, Austria, 2019).

[20] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.108.024308 for additional tables and network visualizations for the four STRING organisms.